

Costruzione di Interfacce Lezione 29 Avanzi

cignoni@iei.pi.cnr.it
<http://vcg.iei.pi.cnr.it/~cignoni>

Introduzione

- ❖ Collision Detection
- ❖ Stencil buffer
- ❖ Rendering to texture
- ❖ Fractal Terrains

Collision Detection

- ❖ La gestione delle collisioni rientra nel più generale problema di simulare la *fisica* del nostro environment ad un adeguato livello di accuratezza:
- ❖ livello 1:
 - ❖ Assicurarsi che il personaggio guidato dal giocatore non attraversi liberamente tutta la scena
- ❖ livello 2:
 - ❖ Tutti gli oggetti mobili della scena si interagiscono correttamente con l'ambiente
- ❖ livello 3:
 - ❖ Tutti gli oggetti mobili interagiscono correttamente anche tra di loro.

Collision Detection: Livello 1

- ❖ Distinzione a livello di scene graph:
 - ❖ tra ambiente e giocatore
 - ❖ Distinzione tra ambiente da controllare e ambiente visualizzato
 - ❖ Il numero di controlli che si fa per frame è lineare con la grandezza della scena
 - ❖ Ottimizzando diventa logaritmico

Approssimare!

- ❖ Il giocatore con una sfera
 - ❖ Nella maggior parte dei casi la differenza non si nota.
- ❖ L'ambiente da testare con pochi poligoni (o con poche sfere)
- ❖ Il test sfera poligono è abbastanza semplice

Vincolare!

- ❖ Ridurre se possibile il problema a 2 dimensioni
 - ❖ Ad esempio in molti giochi first person perspective o racing
- ❖ Ridurre la libertà di movimento
 - ❖ Lo spazio dove si muove il player non è quello cartesiano ma è un insieme discreto di posizioni (e.g. in tutti i giochi di labirinto tipo pacman)

Formule di base

- ❖ Piano (o meglio semispazio)
 - ❖ \mathbf{n} Normale al piano
 - ❖ d distanza dall'origine
- ❖ Per i punti x sul piano vale:
 - ❖ $\mathbf{n} \cdot \mathbf{x} = d$
- ❖ Distanza (con segno) punto p piano pl
 - ❖ $pl.n \cdot p - pl.d$
- ❖ Proiezione p' di un punto p sul piano pl
 - ❖ $k = pl.n \cdot p - pl.d$
 - ❖ $p' = p - pl.n * k$

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

7

Formule

- ❖ Con le formule precedenti è facile sapere il punto più vicino (di contatto?) tra una sfera ed un piano.
- ❖ Per passare a poligoni (triangoli) basta fare il test punto in poligono
- ❖ Molti metodi diversi più o meno efficienti, il più facile da implementare:
 - ❖ la somma degli angoli dal punto ai vertici del poligono è 2π se e solo se sono dentro al poligono.



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

8

Intersezione raggio triangolo

- ❖ Il metodo precedente non è il migliore
- ❖ In effetti quello che spesso serve è il test di intersezione raggio triangolo
- ❖ Questo perchè si vuole sapere se nel prossimo intervallo di tempo lungo la traiettoria corrente colpisce qualcosa.
- ❖ Il metodo che trovate implementato qui è uno dei più efficienti
 - ❖ Tomas Möller and Ben Trumbore:
"Fast, Minimum Storage Ray-Triangle Intersection"
<http://www.acm.org/jgt/papers/MollerTrumbore97/>

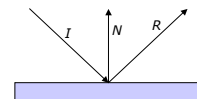
13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

9

Calcolo della risposta

- ❖ Una volta che sappiamo che abbiamo colpito una superficie dobbiamo calcolare la risposta.
- ❖ Il minimo: $R = I - 2 * (I \cdot N) * N$



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

10

Ottimizzazioni

- ❖ Uso di gerarchie di bounding objects semplici (sfere o box) che approssimino in maniera sempre migliore gli oggetti da testare



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

11

Collision e Scene Graph

- ❖ In genere conviene integrare tutto assieme. Ogni nodo che contiene geometria (compresi i gruppi) dovrebbe saper rispondere, efficientemente, alle seguenti query
 - ❖ Bound object
 - ❖ Usato per costruire un bound object per ogni gruppo e limitare la discesa nelle gerarchie
 - ❖ Collision con un punto/sfera/raggio
 - ❖ Restituendo punto di collisione e normale

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

12

Collision e Scene Graph

- ❖ In questo modo si sfrutta la gerarchia dello scene graph e si permette ai singoli nodi di sfruttare la propria conoscenza interna per rispondere efficientemente.
- ❖ E.g. il nodo "anello di moebius" potrebbe calcolare la risposta alle varie query in maniera analitica esatta...

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

13

Stencil Buffer

- ❖ Per-pixel test, simile a depth buffering.
- ❖ Test contro un valore dello stencil buffer; il frammento viene gettato se il test stencil fallisce.
- ❖ Differenti operazioni (aritmetiche!) sullo stencil buffer a seconda che \ul>- ❖ Stencil test fails
- ❖ Depth test fails
- ❖ Depth test passes

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

14

Stencil

- ❖ glEnable/glDisable(GL_STENCIL_TEST);
- ❖ glClear(... | GL_STENCIL_BUFFER_BIT)
- ❖ glStencilFunc(...)
- ❖ glStencilOp(...)
- ❖ glStencilMask(mask)

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

15

Stencil

- ❖ void glStencilFunc(*func*, *GLint ref*, *mask*);
 - ❖ **Setta la funzione e il valore di riferimento con cui fare lo stencil test, il tipo di test è come quello sullo z (NEVER, ALWAYS, LESS, LEQUAL, GREATER ecc)**
 - ❖ **Mask è usata in and sia per il valore sullo stencil buffer che per ref**
- ❖ void glStencilOp(*fail*, *zfail*, *zpass*);
 - ❖ **Le possibili operazioni sono**
 - ❖ Increment, Decrement (saturates)
 - ❖ Keep, Replace
 - ❖ Zero, Invert

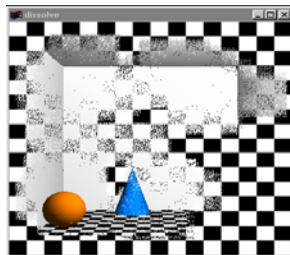
13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

16

Stencil per dissolvenze

- Stencil buffer tiene il pattern di dissolvenza.
- Si disegna due scene cambiando lo stencil test tra una scena e l'altra



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

17

Stencil per riflessioni

- ❖ Riflessioni planari
- ❖ Basta invertire la geometria rispetto al piano dello specchio.
- ❖ Problema: il riflesso esce dallo specchio

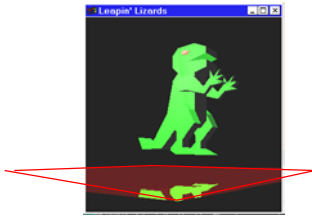


13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

18

Stencil per riflessioni



- ❖ Clear stencil to zero.
Draw floor polygon with stencil set to one.
Only draw reflection where stencil is one

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

19

Stencil Shadow Volume

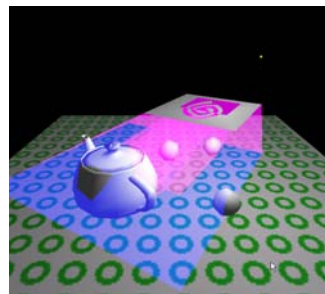
- ❖ Meccanismo di base:
- ❖ Disegnare il volume proiettato da un oggetto che fa ombra in modo tale da capire, usando lo stencil, se un pixel appartiene ad una superficie in ombra o no.

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

20

Stencil Shadow Volume



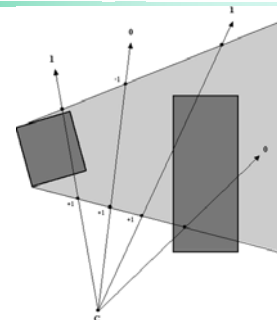
13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

21

Stencil Shadow Volume

- ❖ Lo stencil buffer è **incrementato** quando le front face dello shadow volume passano il depth test.
- ❖ Lo stencil buffer è **decrementato** quando le backface dello shadow volume passano il depth test.
- ❖ I numeri alla fine dei raggi rappresentano i valori lasciati sullo stencil buffer.



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

22

Shadow Volume Algorithm

- ❖ Step by step
 1. Clear frame,depth e stencil
 2. Render scena con luce ambiente
 3. Per ogni oggetto che deve far ombra calcolare la silhouette rispetto alla luce e costruire lo shadow volume
 4. Disegnare le frontface dello shadow volume (senza toccare lo z) incrementando lo stencil
 5. Disegnare le backface dello shadow volume (senza toccare lo z) decrementando lo stencil
 6. Disegnare la scena illuminata non toccando le zone con stencil not zero.

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

23

Fractal Terrains

- ❖ Concetto di base
 - ❖ Selfsimilarity



- ❖ Tecniche principali:
 - ❖ Midpoint displacemnet
 - ❖ Sintesi per somma di funzioni

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

24

MidPoint Displacement

- ❖ In 2d:
- ❖ Partire da un segmento orizzontale (a,b)
- ❖ While(segmento abbastanza grande)
 - ❖ Trova il punto di mezzo p
 - ❖ $H = \text{randomvalue}(-1,1)$
 - ❖ Calcola range spostamento r
 - ❖ Sposta il p verso l'alto di $h*r$
 - ❖ Ricorsivamente su (a,p) e (p,b)



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

25

MidPoint displacement

- ❖ Come si decide il range di spostamento:
- ❖ Sia H un fattore di roughness [0,1]
- ❖ Ad ogni livello di ricorsione
 - ❖ $r = r * 2^{-H}$



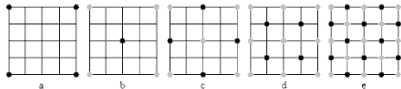
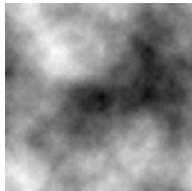
13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

26

Diamond-square alg

- ❖ In 3d un terreno è un height field;
Partire da un grigliato regolare $2^n+1 \times 2^n+1$
- ❖ Si inizia dagli angoli
- ❖ alternativamente si considera quadrati e rombi
- ❖ si sceglie 4 vertici da mediare assieme per trovare la posizione di partenza cui sommare il random displacement



13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

27

Alcuni hint

- ❖ Usare una griglia $2^n \times 2^n$ (anziché $2^n+1 \times 2^n+1$) e usare aritmetica modulo 2^n per fare un terreno tileable
- ❖ Variare H in funzione dell'altezza
 - ❖ I picchi sono rough
 - ❖ Le valli sono smooth
- ❖ Colorare il terreno in funzione dell'altezza (blu, verde, marrone, bianco)
- ❖ Si può usare una texture monodimensionale: $\text{texcoord } s = \text{altezza} + \text{small random}$.

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

28

Rendering to texture

- ❖ Tre meccanismi
- ❖ Opengl 1.0
 - ❖ Rendering nel backbuffer
 - ❖ `glReadPixel ()`
 - ❖ `glTexImage2d()`
 - ❖ Lento, tutto deve passare obbligatoriamente dal processore. Rischio di swizzle di `rgba argb` ecc.

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

29

Rendering to texture

- ❖ Opengl 1.1
 - ❖ Rendering nel backbuffer e `glCopyTexImage2D()`
 - ❖ Efficiente ma siamo vincolati al proprio frame buffer.
 - ❖ risoluzione e depth sono fortemente constrained
 - ❖ Non posso usarlo a metà di un rendering

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

30

Rendering to texture

- ❖ OpenGL arb extension
 - ❖ Pbuffers e render to texture
 - ❖ Estensione wgl che permette di creare contesti non legati a finestre da usare appositamente per off-screen rendering
 - ❖ Indipendenti dalla risoluzione e profondità correnti dello schermo
 - ❖ Path ottimizzati.

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

31

Esempio

Modifichiamo il primo Moebius quello glut, creiamoci un indice di texture ti e aggiungiamo in fondo alla draw:

```
glEnable(GL_TEXTURE_2D);
glBindTexture(GL_TEXTURE_2D,ti);
glColor3f(1,1,1);
glTexEnvi(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
glCopyTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, 0, 0, 512, 512, 0);
glBegin(GL_QUADS);
    glTexCoord2f(0,0);    glVertex3f(-1,-1,0);
    glTexCoord2f(1,0);    glVertex3f(1,-1,0);
    glTexCoord2f(1,1);    glVertex3f(1,1,0);
    glTexCoord2f(0,1);    glVertex3f(-1,1,0);
glEnd();
glDisable(GL_TEXTURE_2D);
```

13 Dicembre 2002

Costruzione di Interfacce - Paolo Cignoni

32